REMARKS/ARGUMENTS

Prior to this Amendment, claims 1-44 were pending.

In this Amendment, claim 1 is amended to clarify that the persistent object framework (POF) is provided logically between a client application and a data source to provide a uniform interface for such client applications to a data source storing persistent objects and to operate a cache for one or more client applications in a transparent manner to the client application. Dependent claim 3 is amended to clarify that the POF acts to create a new persistent object that is cached immediately but not stored in the data source until a save transaction has been committed or a flush method has been invoked to increase data storage and retrieval efficiency. Dependent claim 5 is amended to clarify the POF defers at least some of the updates to persistent objects to improve system efficiencies. Dependent claim 7 is amended to clarify that the mapping performed by the POF is based on a type definition for persistent object and is to two or more data sources.

Independent claims 12, 35, and 36 are amended to clarify similar to claim 3 that creation of a new persistent object by the POF includes deferring storing the new object in the data source until save transaction is committed or flush method is invoked. Independent claims 22, 37, and 38 are amended to clarify how the searching is performed and that the query type is selected from a group of queries not shown by the cited references and the cache is not searched when a filter query is used.

No new matter is added by these amendments with support found at least in Figures 1, 3, and 8 and the specification at page 13, lines 15-20, page 17, lines 19-26, page 37, lines 14-21, and page 69, lines 6-16. Claims 13, 25-29, 31, 32, 39, and 40 are canceled.

Claims 1-12, 14-24, 30, 33-38, and 41-44 remain for consideration by the Examiner.

Claim Rejections Under 35 U.S.C. §112, Second Paragraph

In the February 26, 2004 Office Action, claims 23 and 24 were rejected under 112, second paragraph as being indefinite due to a lack of antecedent basis in claim 23. Claim 23 is amended to provide proper antecedent basis. Claim 24 now also has proper antecedent basis as the rejection was based on its dependency from claim 23.

Claim Rejections Under 35 U.S.C. §102

In the February 26, 2004 Office Action, claims 1-7 were rejected under 102(b) as being anticipated by U.S. Pat. No. 6,035,303 ("Baer"). This rejection is traversed based on the claim amendments and the following remarks.

The present invention, as stated at page 11, line 15, is to provide a persistent object framework (POF) that provides access to persistent objects. The POF preferably functions to "minimize the need for application developers to perform certain tasks" including managing persistent object transactions, managing data source connections, composing query statements, mapping query results to persistent objects and mapping back for data source inserts, updates, and deletes. As stated in the paragraph beginning on page 13, line 15, the POF provides a uniform, simple application program interface for creating, updating, deleting, and querying the persistently stored objects in one or more data sources. To this end, the POF provides features such as object relationships, caching, and deferred writing.

As amended, claim 1 is directed to a system for managing persistent objects. The system comprises a persistent object framework (POF) between an application and a data source. The data source is storing persistent objects and the POF is adapted for providing an interface for the application to the data source. The system further comprises a cached set of persistent objects "within said persistent object framework" and managed by the POF. The cached set includes persistent objects from the data source identified for use by the application.

Because each of these features of the system is not shown or suggested by Baer, claim 1 is not anticipated by Baer.

More particularly, Baer is directed toward a method of storing and accessing persistent objects stored in a digital library. Figures 3A-3C show a good summary of the Baer method, which includes storing objects that are KVDs (key value dictionary which is a subset of a dictionary, see col. 2, line 46)and associating keys (which are "a structural type", see col. 2, line 51) with the KVDs to provide rapid searching based on an index of such keys. A discussion of a prior art system in Figure 1 shows object stores 150, 151 storing persistent objects, a library catalog 140 for such object stores 150, 151, and library clients130, 131. The library clients 130, 131 each must interface with the object server 120 and the library server 110 to query the object stores 150, 151 and must create and manage their own client caches 160. In contrast, claim 1 calls for a POF to be logically positioned between the application and the data source to provide a uniform interface. Further, a cached set of persistent objects identified for use by the application is stored in the POF and managed by the POF, so that the application does not have to perform this function. Hence, the system of Figure 1 in Baer does not anticipate the system of claim 1.

Figure 2 of Baer and corresponding text is also cited as teaching the features of claim 1. However, as can be seen, there is no POF or similar interface structure between the digital library 270 and the application 240. Instead, Java/DL classes 230, JAVA/C++ adapters 250, and the like are provided that must be implemented by the application 240 (rather than by a POF). The object vault 210 is said to contain a repository of APIs at col. 5, line 5 but apparently these API can be selectively implemented by the application which is shown to be able to interact directly with the digital library 270. More significantly, the object vault 210 is not taught to manage a cache of persistent objects identified for use by the application 240. Hence, when the prior art of Figure 1 is combined with the Baer teaching of Figure 2, any caching of persistent objects would need to be performed by and managed by the client application 240 not the object vault 210, which functions to create an index of blobs or KVDs in the library 270 and to facilitate searching. For these reasons, claim 1 is believed allowable over Baer.

Claims 2-7 depend from claim 1 and are allowable at least for the reasons for allowing claim 1. Additionally, claim 3 calls the POF to create a persistent object, to cache the new object, and then to delay storage of the persistent object until <u>after a save transaction has been committed or a flush method has been invoked</u>, which is not shown or suggested by Baer as the object vault creates indexes but does not create, cache, and later store new objects. Claim 5 calls for the POF to update a persistent object including <u>deferring a write until a transaction is committed, a flush method is invoked, or a query process is started</u>. Baer does not discuss writing or updating stored persistent objects except in passing and clearly, does not indicate the usefulness of deferring such updating. Claim 7 calls for the POF to provide mapping of persistent objects based on their defined type to two or more data sources. Baer fails to teach such mapping. For these additional reasons, claims 3, 5, and 7 are allowable over Baer.

Additionally, in the Office Action, independent claims 33, 41, and 42 were rejected under 102(b) as being anticipated by U.S. Pat. No. 6,065,013 ("Fuh"). This rejection is traversed based on the following remarks.

Claim 33 is directed to a method for resolving a stale data state between a persistent object and an application access the object for data. The method comprises accessing with an application a persistent object that includes "a revision attribute." A stale data state is then identified within the persistent object by a POF. The method includes "retrying said process of said application accessing said persistent object" and "incrementing the revision attribute." Applicants have reviewed Fuh and cannot find any of these features in the portions cited by the Office Action or elsewhere. Hence, claim 33 is not anticipated by Fuh.

Specifically, the Office Action cites the Abstract, Figure 5, col. 2, lines 5-20, col. 4, lines 53 to col. 5, line 3, and col. 8, lines 11-26 for teaching the executing process including a persistent object with a revision attribute. However, Fuh fails to teach including a revision attribute in any of these citations (see, for example, Figure 5 or col. 8, lines 11-26). Further, Fuh does not teach identifying a stale data state within the persistent object with a POF, retrying the process, or incrementing the revision attribute. Fuh is directed toward implementing storage of persistent objects and using an inline buffer for storing the objects within a database, but Applicants could find no discussion of stale data within such objects or retrying a process in an application that is trying to access identified stale data. As Fuh does not teach the objects having a revision attribute, it cannot teach incrementing the attribute. Hence, claim 33 is not anticipated or even suggested by the very different teaching of Fuh.

Independent claims 43 and 44 are directed to a system and a computer program product with very different claim language than claim 33. Hence, Applicants believe that the grouping of the claims 43 and 44 with claim 33 was a mistake (also, the Office Action states the limitations were discussed with relation to claim 12 which has not yet been discussed and which includes a "mapping" process not included in claims 43 and 44). Specifically, claims 43 and 44 do not discuss the use of revision attributes in managing stale data states within persistent objects or retrying accessing processes within an application as called for in claim 33. However, claims 43 and 44 are believed allowable over Fuh for at least the reason that Fuh does not teach "deferring writes to said first and second data sources" or "caching said persistent objects."

Further, in the February 26, 2004 Office Action, claims 31, 32, 34, 39, 40, 43, and 44 were rejected under 102(e) as being unpatentable over U.S. Pat. No. 6,453,321 ("Hill"). Claims 31, 32, 39, and 40 are canceled. The rejection of claims 34, 40, and 43 is traversed based on the following remarks.

Claims 34, 40, and 43 in varying form include the limitation that a POF is implemented to cache persistent objects from first and second data sources including "deferring writes to said first and second data sources." Hill fails to teach a POF caching objects from 2 data sources. Also, Hill fails to teach the usefulness of deferring writes to the first and second data sources. Hill is cited at col. 5, lines 1-9 for teaching such write deferring but this citation appears to be merely discussing using associations to facilitate searching a cache and searching a database if a cache search fails. Applicants can find even a suggestion that writes to a data source should be deferred. Hence, claims 34, 40, and 43 are not anticipated by the teaching of Hill.

Claim Rejections Under 35 U.S.C. §103

In the February 26, 2004 Office Action, claims 8-21, 35, and 36 were rejected under 103(a) as being unpatentable over Baer in view of U.S. Pat. Pub. No. 2002/0147857 ("Sanchez"). This rejection is traversed based on the following remarks.

Claim 8 is directed to an application system that comprises both a relational database and a LDAP repository each storing a set of persistent objects. The system further comprises a POF to provide data to an application from the sets of objects stored in the 2 differing data sources. Further, the POF caches a subset of each of the sets of persistent objects. In other words, the POF acts as an interface to 2 differing data sources and acts to cache objects from both sets. This is not taught by the combination of Baer and Sanchez, and claim 8 is allowable over this combination.

As discussed with reference to claim 1, Baer fails to teach the use of a POF as an interface to a data source. Baer also fails to teach a POF or similar mechanism for creating and managing a cache of objects such that the application does not have to perform such a function (can be ignorant of the configuration or even type of data source storing the persistent objects). The Office Action confirms that Baer does not teach a LDAP repository for storing a second set of persistent objects correlating to an application or a POF for providing data from sets cached from 2 differing data sources for an application. Hence, the Office Action cites Sanchez in an attempt to overcome this deficiency.

However, there is no motivation in Baer to utilize 2 differing data sources concurrently or to provide an interface for an application between such data sources (e.g., see Figures 1 and 2 where no interface is provide and/or where no caching is provided by a service between the application and the data sources). Sanchez simply teaches that a LDAP data source may be used to store persistent objects but fails to overcome the other deficiencies of Baer. Combining Baer and Sanchez would result in the use of a single data source for the digital library of Figures 1 and 2 of Baer that is an LDAP repository and possibly a cache run by the application, but the combination would not result in the invention of claim 8 with 2 differing data sources, a POF providing a uniform interface for applications, and a cache within the POF of sets of persistent objects from the 2 differing data sources. Claims 9-11 depend from claim 8 and are believed allowable as depending from an allowable base claim. For these reasons, claims 8-11 are allowable over the combination of Baer and Sanchez.

As amended, independent claim 12 is directed to a method of managing persistent objects that includes caching persistent objects within a POF and creating a new object with the POF including <u>caching the new persistent object in the persistent object cache and inserting the new persistent object in the data source after a save transaction has been committed or a flush method has been invoked</u>. As discussed with reference to claim 1, Baer fails to teach a POF that manages a cache for an application, and as discussed with reference to claim 3, Baer fails to teach that a POF that creates new persistent objects by first caching the object and then after save transaction is committed or a flush method invoked the new object is inserted into the data source. Sanchez does not overcome the deficiencies of Baer, and apparently is only cited for its teaching of a mapping method. As a result, claim 12 and claims 13-21 that depend from claim 12 are non-obvious in view of this combination of references. Independent claims 35 and 36 are directed to a system and a computer program product, respectively, with similar limitations to that of claim 12, and the reasons for allowing claim 12 are equally applicable to claims 35 and 36.

Further, in the Office Action, claims 22-30, 37, and 38 were rejected under 103(a) as being unpatentable over Fuh in view of Hill. Claims 25-29 are canceled. The rejection of claims 22-23, 30, 37, and 38 is traversed based on the amendments to the claims and the following remarks.

Independent claims 22, 37, and 38 in varying claim format call for receiving a search query for a persistent object and then determining a query type for the search query. The query type may be selected from the group consisting of <u>a primary key, a handle, a unique key, a query filter, and a relationship between persistent objects</u>. When the query type is a query filter, then the search is performed first of a cache of such persistent objects <u>within a POF</u> and the data source is only searched if an object is not found in the cache or the query type is a query filter type. Because each of these features is not taught or suggested by the combination of Fuh and Hill, independent claims 22, 37, and 38 are in condition for allowance.

Fuh is said to teach everything but the searching by query type. However, Fuh fails to teach creating and managing a cache of persistent objects by a POF or similar mechanism. Hence, the combination of Fuh and Hill cannot teach or suggest each and every element of claims 22, 37, and 38. Further, Hill is cited for teaching searching by each of the query types of claims 22, 37, and 38. At best, Hill appears to teach in Figures 7 and 8 and corresponding text searching a database of objects based on keys. However, Hill fails at least to teach the use of query types that include filters. The Office Action cites Figures 6 and 8 but neither of these illustrated processes of Hill discuss a search by filter (as shown in Figure 5 block 502 by Applicants). Further, if the search is a filter-type query, then the cache is not searched but instead the search proceeds directly to the data source. This conditional feature of claims 22, 37, and 38 is not shown by Hill. Hence, claims 22, 37, and 38 are allowable over Fuh and Hill. Claims 23-30 depend from claim 22 and are believed allowable as depending from an allowable base claim.

Conclusion

The references made of record in the February 26, 2004 Office Action but not relied upon by the Examiner have been considered by Applicants. These references are not believed any more relevant than those relied upon, and the pending claims are believed allowable over these additional references.

In view of all of the above, the claims are now believed to be allowable and the case in condition for allowance which action is respectfully requested. Should the Examiner be of the opinion that a telephone conference would expedite the prosecution of this case, the Examiner is requested to contact Applicants' attorney at the telephone number listed below.

No fee is believed due for this submittal. However, any fee deficiency associated with this submittal may be charged to Deposit Account No. 50-1123.

Respectfully submitted,

May 4, 2004

[illegible]

Kent A. Lembke, No. 44,866
Hogan & Hartson LLP
One Tabor Center
1200 17th Street, Suite 1500
Denver, Colorado 80202
(720) 406-5378 Tel
(303) 899-7333 Fax